In value-based deep reinforcement learning, a pruned network is a good network

Johan Obando-Ceron [1 2 3] **Aaron Courville** [2 3] **Pablo Samuel Castro** [1 2 3]

Abstract

Recent work has shown that deep reinforcement learning agents have difficulty in effectively using their network parameters. We leverage prior insights into the advantages of sparse training techniques and demonstrate that gradual magnitude pruning enables value-based agents to maximize parameter effectiveness. This results in networks that yield dramatic performance improvements over traditional networks, using only a small fraction of the full network parameters. **Our code is publicly available,** see Appendix A for details.

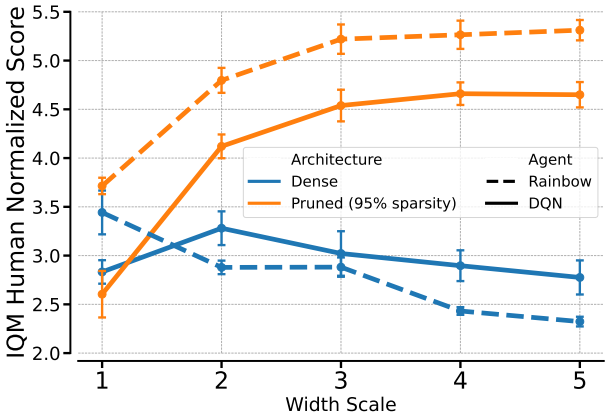

Figure 1. **Scaling network widths for ResNet architecture**, for DQN and Rainbow with an Impala-based ResNet (Espeholt et al., 2018). We report the interquantile mean after 40 million environment steps, aggregated over 15 games with 5 seeds each; error bars indicate 95% stratified bootstrap confidence intervals. Replay ratio is fixed to the standard 0.25. The default network is *Dense*, which we indicate with a **blue** color in all the plots, for clarity.

1. Introduction

Despite successful examples of deep reinforcement learning (RL) being applied to real-world problems (Mnih et al., 2015; Berner et al., 2019; Vinyals et al., 2019; Fawzi et al., 2022; Bellemare et al., 2020), there is growing evidence of challenges and pathologies arising when training these networks (Ostrovski et al., 2021; Kumar et al., 2021a; Lyle et al., 2022; Graesser et al., 2022; Nikishin et al., 2022; Sokar et al., 2023; Ceron et al., 2023). In particular, it has been shown that deep RL agents *under-utilize* their network's parameters: Kumar et al. (2021a) demonstrated that there is an implicit underparameterization, Sokar et al. (2023) revealed that a large number of neurons go dormant during training, and Graesser et al. (2022) showed that sparse training methods can maintain performance with a very small fraction of the original network parameters.

One of the most surprising findings of this last work is that applying the gradual magnitude pruning technique proposed by Zhu & Gupta (2017) on DQN (Mnih et al., 2015) with a ResNet backbone (as introduced in Impala (Espeholt et al., 2018)), results in a 50% performance improvement over the dense counterpart, with only 10% of the original parameters (see the bottom right panel of Figure 1 of Graesser et al. (2022)). Curiously, when the same pruning technique is applied to the original CNN architecture there are no performance improvements, but no degradation either.

That the same pruning technique can have such qualitatively different, yet non-negative, results by simply changing the underlying architecture is interesting. It suggests that training deep RL agents with non-standard network topologies (as induced by techniques such as gradual magnitude pruning) may be generally useful, and warrants a more profound investigation.

In this paper we explore gradual magnitude pruning as a general technique for improving the performance of RL agents. We demonstrate that in addition to improving the performance of standard network architectures for value-based agents, the gains increase proportionally with the size of the base network architecture. This last point is significant, as deep RL networks are known to struggle with scaling architectures (Ota et al., 2021; Farebrother et al., 2023; Taiga et al., 2023; Schwarzer et al., 2023).

[*]Equal contribution [1]Google DeepMind [2]Mila - Québec AI Institute [3]Université de Montréal. Correspondence to: Johan Obando-Ceron <jobando0730@gmail.com>, Pablo Samuel Castro <psc@google.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

Our main contributions are as follows. We:

- present gradual magnitude pruning as a general technique for maximizing parameter efficiency in value-based RL;
- demonstrate that networks trained with this technique produce stronger agents than their dense counterparts, and continue improving as we scale the size of the initial network;
- investigate this technique across a varied set of agents and training regimes, including actor-critic methods;
- present in-depth analyses to better understand the reasons behind their benefits.

2. Related Work

Scaling in Deep RL Deep neural networks have been the driving factor behind many of the successful applications of reinforcement learning to real-world tasks. However, it has been historically difficult to scale these networks, in a manner similar to what has led to the "scaling laws" in supervised learning, without performance degradation; this is due in large part to exacerbated training instabilities that are endemic to reinforcement learning (Van Hasselt et al., 2018; Sinha et al., 2020; Ota et al., 2021). Recent works that have been able to do so successfully have had to rely on a number of targeted techniques and careful hyperparameter selection (Farebrother et al., 2023; Taiga et al., 2023; Schwarzer et al., 2023; Ceron et al., 2023).

Cobbe et al. (2020); Farebrother et al. (2023) and Schwarzer et al. (2023) switched from the original CNN architecture of Mnih et al. (2015) to a ResNet based architecture, as proposed by Espeholt et al. (2018), which proved to be more amenable to scaling. Cobbe et al. (2020) and Farebrother et al. (2023) observe advantages when increasing the number of features in each layer of the ResNet architecture. Schwarzer et al. (2023) show that the performance of their agent (BBF) continues to grow proportionally with the width of their network. Bjorck et al. (2021) propose spectral normalization to mitigate training instabilities and enable scaling of their architectures. Ceron et al. (2023) propose reducing batch sizes for improved performance, even when scaling networks.

Ceron et al. (2024) demonstrate that while parameter scaling with convolutional networks hurts single-task RL performance on Atari, incorporating Mixture-of-Expert (MoE) modules in such networks improves performance. Farebrother et al. (2024) demonstrate that value functions trained with categorical cross-entropy significantly improves performance and scalability in a variety of domains.

Sparse Models in Deep RL Previous studies (Schmitt et al., 2018; Zhang et al., 2019) have employed knowledge distillation with static data to mitigate instability, resulting in small, but dense, agents. Livne & Cohen (2020) introduced policy pruning and shrinking, utilizing iterative policy pruning similar to iterative magnitude pruning (Han et al., 2015), to obtain a sparse DRL agent. The exploration of the lottery ticket hypothesis in DRL was initially undertaken by Yu et al. (2019), and later Vischer et al. (2021) demonstrated that a sparse winning ticket can also be identified through behavior cloning. Sokar et al. (2021) proposed the use of structural evolution of network topology in DRL, achieving 50% sparsity with no performance degradation. Arnob et al. (2021) introduced single-shot pruning for offline Reinforcement Learning.

Graesser et al. (2022) discovered that pruning often yields improved results, and *dynamic* sparse training methods, where the sparse topology changes throughout training (Mocanu et al., 2018; Evci et al., 2020), can significantly outperform *static* sparse training, where the sparse topology remains fixed throughout training. Tan et al. (2023) enhance the efficacy of dynamic sparse training through the introduction of a novel delayed multi-step temporal difference target mechanism and a dynamic-capacity replay buffer. Grooten et al. (2023) proposed an automatic noise filtering method, which uses the principles of dynamic sparse training for adjusting the network topology to focus on task-relevant features.

Overparameterization in Deep RL Song et al. (2019) and Zhang et al. (2018) highlighted and the tendency of RL networks to overfit, while Nikishin et al. (2022) and Sokar et al. (2023) demonstrated the prevalence of plasticity loss in RL networks, leading to a decline in final performance. Several strategies have been proposed to mitigate this, such as data augmentation (Yarats et al., 2021; Cetin et al., 2022), dropout (Gal & Ghahramani, 2016), and layer and batch normalization (Ba et al., 2016; Ioffe & Szegedy, 2015). Hiraoka et al. (2021) demonstrated the success of employing dropout and layer normalization in Soft Actor-Critic (Haarnoja et al., 2018), while Liu et al. (2020) identified that applying ℓ_2 weight regularization on actors can enhance both on- and off-policy algorithms.

Nikishin et al. (2022) identify a tendency of networks to overfit to early data (the primacy bias), which can hinder subsequent learning, and propose periodic network re-initialization as a means to mitigate it. Similarly, Sokar et al. (2023) proposed re-initializing *dormant neurons* to improve network plasticity, while Nikishin et al. (2023) propose plasticity injection by temporarily freezing the current network and utilizing newly initialized weights to facilitate continuous learning.

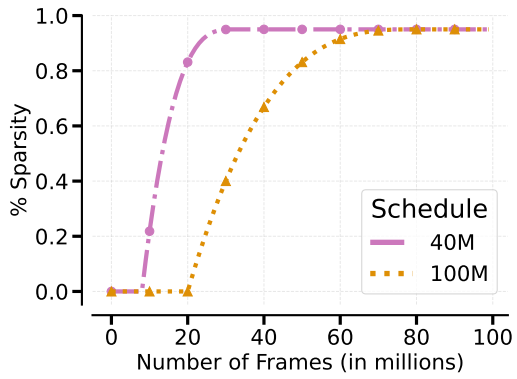

Figure 2. **Gradual magnitude pruning schedules** used in our experiments, to a target sparsity of 95%, as specified in Equation 1. Impact of varying pruning schedules, see Figure 10.

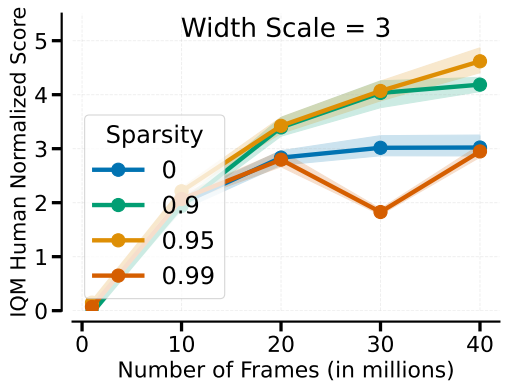

Figure 3. **Evaluating how varying sparsity affects performance** for DQN with the ResNet architecture and a width multiplier of 3. See Section 4.1 for training details.

3. Background

Deep reinforcement learning The goal in Reinforcement Learning is to optimize the cumulative discounted return over a long horizon, and is typically formulated as a Markov decision process (MDP) $(\mathcal{X}, \mathcal{A}, P, r, \gamma)$. An MDP is comprised of a state space $\mathcal{X}$, an action space $\mathcal{A}$, a transition dynamics model $P : \mathcal{X} \times \mathcal{A} \to \Delta(\mathcal{X})$ (where $\Delta(X)$ is a distribution over a set X), a reward function $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, and a discount factor $\gamma \in [0, 1)$. A policy $\pi : \mathcal{X} \to \Delta(\mathcal{A})$ formalizes an agent's behaviour.

For a policy π, $Q^\pi(\mathbf{x}, \mathbf{a})$ represents the expected discounted reward achieved by taking action $\mathbf{a}$ in state $\mathbf{x}$ and subsequently following the policy π: $Q^\pi(\mathbf{x}, \mathbf{a}) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}\left(\mathbf{x}_t, \mathbf{a}_t\right) | \mathbf{x}_0 = x, \mathbf{a}_0 = a\right]$. The optimal Q-function, denoted as $Q^\star(\mathbf{x}, \mathbf{a})$, satisfies the Bellman recurrence
$Q^\star(\mathbf{x}, \mathbf{a}) = \mathbb{E}_{\mathbf{x}' \sim P(\mathbf{x}'|\mathbf{x},\mathbf{a})} \left[\mathcal{R}(\mathbf{x}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q^\star\left(\mathbf{x}', \mathbf{a}'\right)\right]$.

Most modern value-based methods will approximate Q via a neural network with parameters θ, denoted as Q_θ. This idea was introduced by Mnih et al. (2015) with their DQN agent, which has served as the basis for most modern deep RL algorithms. The network Q_θ is typically trained with a *temporal difference loss*, such as:

$$L(\theta) = \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{a},\mathbf{r},\mathbf{x}') \sim \mathcal{D}} \left[\left(\mathbf{r} + \gamma \max_{\mathbf{a}' \in \mathcal{A}} \bar{Q}\left(\mathbf{x}', \mathbf{a}'\right) - Q_\theta(\mathbf{x}, \mathbf{a})\right)^2\right],$$

Here $\mathcal{D}$ represents a stored collection of transitions $(\mathbf{x}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{x}_{t+1})$ which the agent samples from for learning (known as the replay buffer). $\bar{Q}$ is a static network that infrequently copies its parameters from Q_θ; its purpose is to produce stabler learning targets.

Rainbow (Hessel et al., 2018) extended, and improved, the original DQN algorithm with double Q-learning (van Hasselt et al., 2016), prioritized experience replay (Schaul et al., 2016), dueling networks (Wang et al., 2016), multi-step returns (Sutton, 1988), distributional reinforcement learning (Bellemare et al., 2017), and noisy networks (Fortunato et al., 2018).

Gradual pruning In supervised learning settings there is a broad interest in sparse training techniques, whereby only a subset of the full network parameters are trained/used (Gale et al., 2019). This is motivated by computational and space efficiency, as well as speed of inference. Zhu & Gupta (2017) proposed a polynomial schedule for gradually sparsifying a dense network over the course of training by *pruning* model parameters with low weight magnitudes.

Specifically, let $s_F \in [0, 1]$ denote the final desired sparsity level (e.g. 0.95 in most of our experiments) and let t_{start} and t_{end} denote the start and end iterations of pruning, respectively; then the sparsity level at iteration t is given by:

$$s_t = \begin{cases} s_F\left(1 - \left(1 - \dfrac{t - t_{\text{start}}}{t_{\text{end}} - t_{\text{start}}}\right)^3\right) & \text{if } t_{\text{start}} \leq t \leq t_{\text{end}} \\ 0.0 & \text{if } t < t_{\text{start}} \\ s_F & \text{if } t > t_{\text{end}} \end{cases} \tag{1}$$

Graesser et al. (2022) applied this idea to deep RL networks, setting t_{start} at 20% of training and t_{end} at 80% of training.

4. Pruning can boost deep RL performance

We investigate the general usefulness of gradual magnitude pruning in deep RL agents in both online and offline settings.

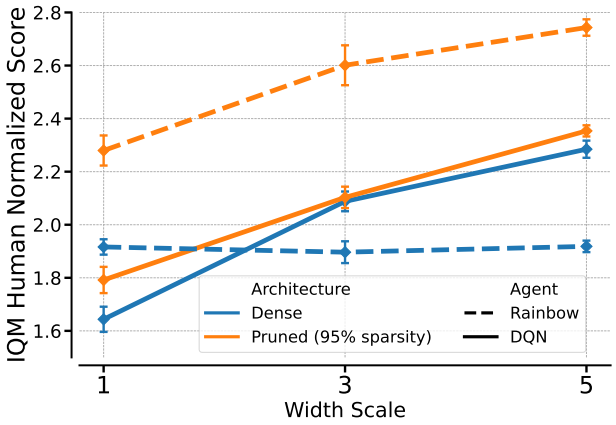

Figure 4. **Scaling network widths for the original CNN architecture of Mnih et al. (2015)**, for DQN **(left)** and Rainbow **(right)**. See Section 4.1 for training details.

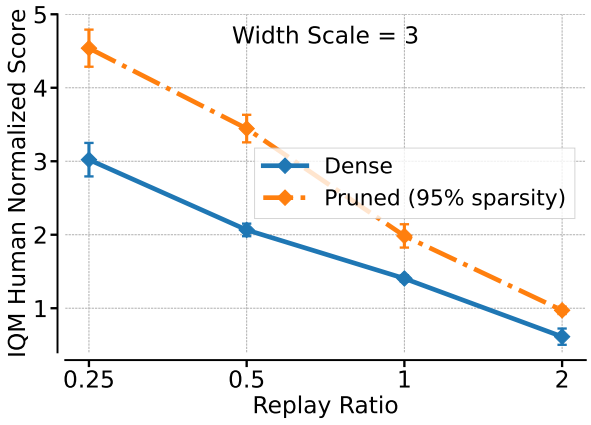

Figure 5. **Scaling replay ratio for Rainbow with the ResNet architecture with a width multiplier of 3.** Default replaly ratio is 0.25. See Section 4.1 for training details.

4.1. Implementation details

For the base DQN and Rainbow agents we use the Jax implementations of the Dopamine library[1] (Castro et al., 2018) with their default values. It is worth noting that Dopamine provides a "compact" version of the original Rainbow agent, using only multi-step updates, prioritized replay, and distributional RL. For all experiments we use the Impala architecture introduced by Espeholt et al. (2018), which is a 15-layer ResNet, unless otherwise specified. Our reasoning for this is not only because of the results from Graesser et al. (2022), but also due to a number of recent works demonstrating that this architecture results in generally improved performance (Schmidt & Schmied, 2021; Kumar et al., 2022; Schwarzer et al., 2023).

We use the JaxPruner[2] (Lee et al., 2024) library for gradual magnitude pruning, as it already provides integration with Dopamine. We follow the schedule of Graesser et al. (2022): begin pruning the network 20% into training and stop at 80%, keeping the final sparse network fixed for the rest of training. Figure 2 illustrates the pruning schedules used in our experiments (for 95% sparsity). We evaluate our agents on the Arcade Learning Environment (ALE) (Bellemare et al., 2013) on the same 15 games used by Graesser et al. (2022), chosen for their diversity[3]. For computational considerations, most experiments were conducted over 40 million frames (as opposed to the standard 200 million); in our investigations we found the qualitative differences between algorithms at 40 million frames to be mostly consistent with those at 100 million (e.g. see Figure 10).

[1]Dopamine code available at `https://github.com/google/dopamine`.

[2]JaxPruner code available at `https://github.com/google-research/jaxpruner`.

[3]Discussed in A.4 in Graesser et al. (2022).

We follow the guidelines outlined by Agarwal et al. (2021) for evaluation: each experiment was run with 5 independent seeds, and we report the human-normalized interquantile mean (IQM), aggregated across the 15 games, configurations, and seeds, along with 95% stratified bootstrap confidence intervals. All experiments were run on NVIDIA Tesla P100 GPUs, and each took approximately 2 days to complete. All hyper-parameters are listed in Appendix F.

4.2. Online RL

While Graesser et al. (2022) demonstrates that sparse networks are capable of maintaining agent performance, if these levels of sparsity were too high, performance eventually degrades. This is intuitive, as with higher levels of sparsity, there are fewer active parameters left in the network. One natural question is whether *scaling* our initial network enables high levels of sparsity. We thus begin our inquiry by applying gradual magnitude pruning on DQN with the Impala architecture, where we have scaled the convolutional layers by a factor of 3. Figure 3 confirms that this is indeed the case: 90% and 95% sparsity produce a 33% performance improvement, and 99% sparsity maintains performance.

A sparsity of 95% consistently yielded the best performance in our initial explorations, so we primarily focus on this sparsity level for our investigations. Figure 1 is a striking result: we observe close to a 60% (DQN) and 50% (Rainbow) performance improvement over the original (unpruned and unscaled) architectures. Additionally, while the performance of the unpruned architectures decreases monotonically with increasing widths, the performance of the pruned counterparts *increases* monotonically. In Figure 6 we evaluated pruning on DQN and Rainbow over all 60 Atari 2600 games, confirming our findings are not specific to the 15 games initially selected.

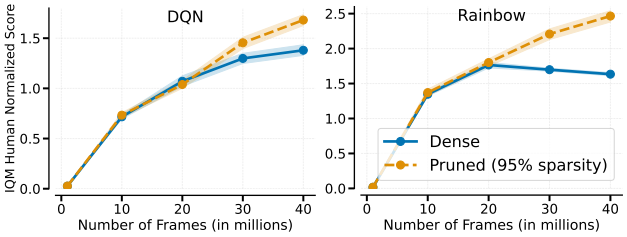

Figure 6. **Evaluating performance on the full Atari 2600 suite.** DQN (left) and Rainbow (right), both using the ResNet architecture with a width of 3. We report IQM performance with error bars indicating 95% confidence interval. See Section 4.1 for training details.

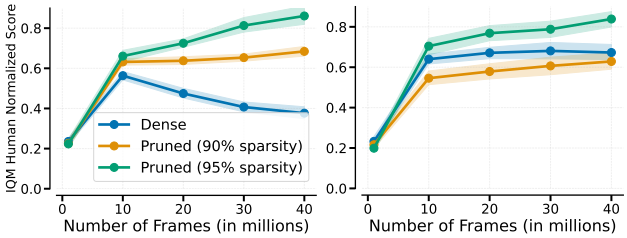

Figure 7. **Performance of DrQ(ϵ) (left) and DER (right) when trained for 40M frames**. Both agents use a ResNet architecture with a width multiplier of 3. See Section 4.1 for training details.

When switching both agents to using the original CNN architecture of Mnih et al. (2015) we see a similar trend with Rainbow, but see little improvement in DQN (Figure 4). This result is consistent with the findings of Graesser et al. (2022), where no real improvements were observed with pruning on CNN architectures. An interesting observation is that, with this CNN architecture, the performance of DQN does seem to benefit from increased width, while the performance of Rainbow suffer from slight degradation.

When evaluating on more modern value-based agents, specifically IQN (Dabney et al., 2018) and Munchausen-IQN (Vieillard et al., 2020), we observe the same advantages arising from pruning (see Appendix G.2).

Our findings thus far suggest that the use of gradual magnitude pruning increases the parameter efficiency of these agents. If so, then these sparse networks should also be able to benefit from more gradient updates. The *replay ratio*[4], which is the number of gradient updates per environment step, measures exactly this; it is well-known that it is difficult to increase this value without performance degradation (Fedus et al., 2020; Nikishin et al., 2022; Schwarzer et al., 2023; D'Oro et al., 2023).

In Figure 5 we can indeed confirm that the pruned architectures maintain a performance lead over the unpruned baseline even at high replay ratio values. The sharper rate of decline with pruning may suggest that the pruning schedule needs to be re-tuned for these settings.

4.3. Low data regime

Kaiser et al. (2020) introduced the Atari 100k benchmark to evaluate RL agents in a sample-constrained setting, allowing agents only 100k[5] environment interactions. For this regime, Kostrikov et al. (2020) introduced DrQ, a variant of DQN which makes use of data augmentation; the hyperparameters for this agent were further optimized by Agarwal et al. (2021) in DrQ(ϵ). Similarly, Van Hasselt et al. (2019) introduced Data-Efficient Rainbow (DER), which optimized the hyperparameters of Rainbow (Hessel et al., 2018) for this low data regime.

When evaluated on this low data regime, our pruned agents demonstrated no gains. However, when we ran for 40M environment interactions (as suggested by Ceron et al. (2023)), we do observe significant gains when using gradual magnitude pruning, as shown in Figure 7. Interestingly, In DrQ(ϵ) the pruned agents avoid the performance degradation affecting the baseline when trained for longer.

4.4. Offline RL

Offline reinforcement learning focuses on training an agent solely from a fixed dataset of samples without any environment interactions. We used two recent state of the art methods from the literature: CQL (Kumar et al., 2020) and CQL+C51 (Kumar et al., 2022), both with the ResNet architecture from Espeholt et al. (2018). Following Kumar et al. (2021b), we trained these agents on 17 Atari games for 200 million frames iterations, where where 1 iteration corresponds to 62,500 gradient updates. We assessed the agents by considering a dataset composed of a random 5% sample of all the environment interactions collected by a DQN agent trained for 200M environment steps (Agarwal et al., 2020).

Note that since we are training for a different number of steps than our previous experiments, we adjust the pruning schedule accordingly. As shown in Figure 8, both CQL and CQL+C51 observe significant gains when using pruned networks, in particular with wider networks. Interestingly, in the offline regime, pruning also helps to avoid performance collapse when using a shallow network (width scale equal to 1), or even improve final performance as in the case of CQL+C51.

[4]In the hyperparameters established in (Mnih et al., 2015), the policy is updated every 4 environment steps collected, resulting in a replay ratio of 0.25.

[5]Here, 100k refers to agent steps, or 400k environment frames, due to skipping frames in the standard training setup.

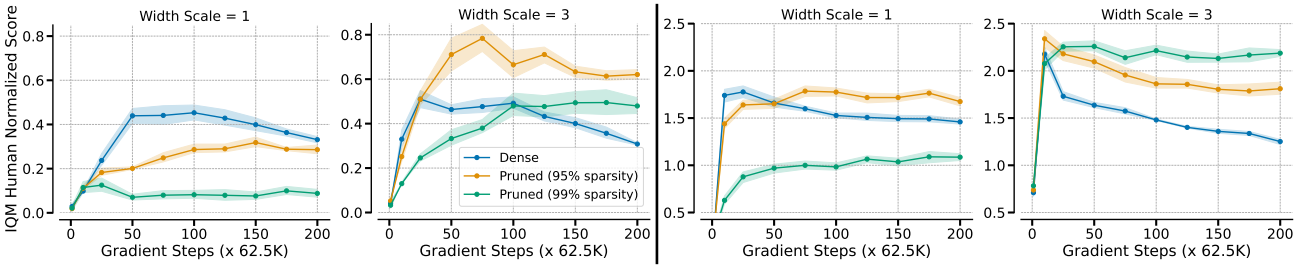

Figure 8. **Scaling network widths for offline agents** CQL **(left)** and CQL+C51 **(right)**, both using the ResNet architecture. We report interquantile mean performance with error bars indicating 95% confidence intervals across 17 Atari games. x-axis represents gradient steps; no new data is collected.

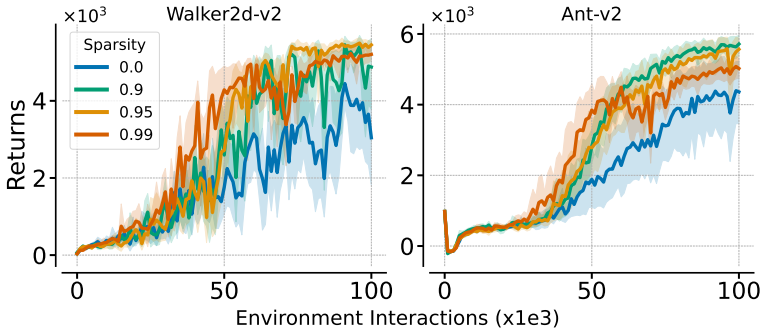

Figure 9. **Evaluating how varying the sparsity parameter affects performance of SAC** on two MuJoCo environments when increasing width x5. We report returns over 10 runs for each experiment.

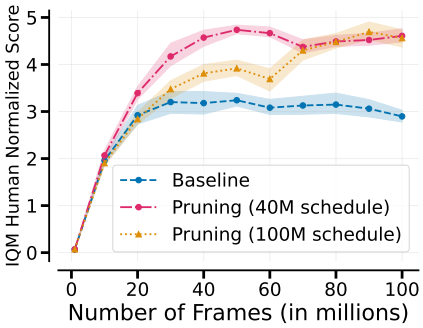

Figure 10. **Impact of varying pruning schedules**, for DQN with an Impala-based ResNet with a width multiplier of 3.

4.5. Actor-Critic methods

Our investigation thus far has focused on value-based methods. Here we investigate if gradual magnitude pruning can yield performance gains for Soft Actor Critic (SAC; Haarnoja et al., 2018), a popular policy-gradient algorithm. We evaluated SAC on five continuous control environments from the MuJoCo suite (Todorov et al., 2012), using 10 independent seeds for each. In Figure 9 we present the results for Walker2d-v2 and Ant-v2, where we see the advantages of gradual magnitude pruning persist; in the remaining three environments (see Appendix E) there are no real observable gains from pruning. In Appendix G.1 we see a similar trend with PPO (Schulman et al., 2017).

4.6. Stability of the pruned network

We followed the pruning schedule proposed by Graesser et al. (2022), which adapts naturally to differing training steps (as discussed above for the offline RL experiments). This schedule trains the final sparse network for only the final 20% of training steps. A natural question is whether the resulting sparse network, when trained for longer, is still able to maintain its performance. To evaluate this, we trained DQN for 100 million frames and applied two pruning schedules: the regular schedule we would use for 100M as well as the schedule we would normally use for 40M training steps (see Figure 2).

As Figure 10 shows, even with the compressed 40M schedule, the pruned network is able to maintains its strong performance. Interestingly, with the compressed schedule the agent achieves a higher performance *faster* than with the regular one. This suggests there is ample room for exploring alternate pruning schedules.

4.7. Learning rate and Batch size scaling

The default learning rate or batch size may not be optimal for large neural networks. The default learning for DQN is 6.25×10^{-5}. We run experiments with a learning rate divided by the width scale factor (so 2.08×10^{-5} for a width factor of 3, and 1.25×10^{-5} for a width factor of 5). While these learning rates do improve the performance of the baseline, it is still surpassed by pruning (see Figure 28). We observe a similar trend when evaluating different batch size values. The default batch size is 32 (for all value based agents used in this paper), and we ran experiments with batch sizes of 16 and 64. In all cases, pruning maintains its strong advantage (see Figure 29). These results are consistent with the thesis that *pruning can serve as a drop-in mechanism for increasing agent performance*.

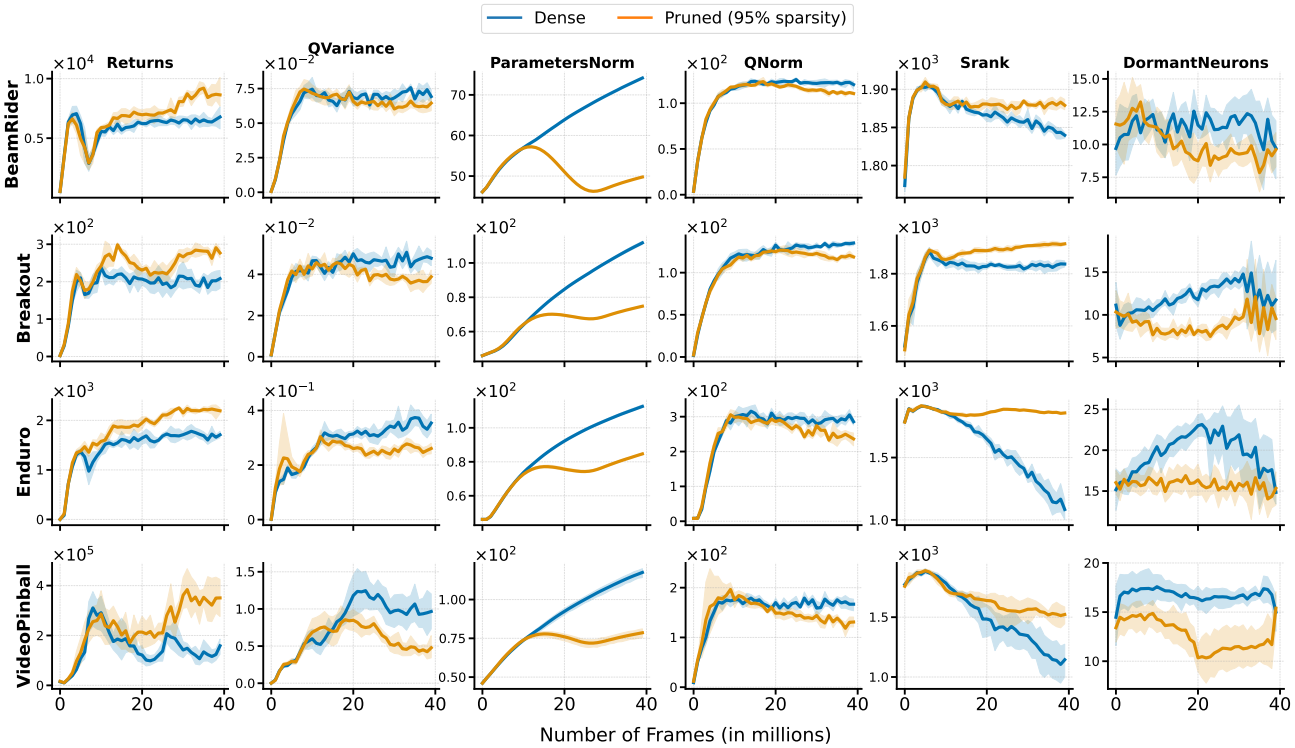

Figure 11. **Empirical analyses for four representative games when applying pruning**. From left to right: training returns, average Q-target variance, average parameters norm, average Q-estimation norm, $srank$ (Kumar et al., 2021a), and dormant neurons (Sokar et al., 2023). All results averaged over 5 seeds, shaded areas represent 95% confidence intervals.

5. Why is pruning so effective?

We focus our analyses on four games: BeamRider, Breakout, Enduro, and VideoPinball. For each, we measure the variance of the Q estimates (QVariance); the average norm of the network parameters (ParametersNorm); the average norm of the Q-values (QNorm); the effective rank of the matrix (Srank) as suggested by Kumar et al. (2021a), and the fraction of dormant neurons as defined by Sokar et al. (2023).

We present our results in Figure 11. What becomes evident from these figures is that pruning **(i)** reduces variance, **(ii)** reduces the norms of the parameters, **(iii)** decreases the number of dormant neurons, and **(iv)** increases the effective rank of the parameters. Some of these observations can be attributed to a form of normalization, whereas others may arise due to increased network plasticity.

Lyle et al. (2024) show that increased parameter norm accompanies plasticity loss in different neural architectures. In Figure 11, we observe a low parameter norm value when applying gradual pruning, which represent a high final performance return.

5.1. Comparison to other methods

In order to disentangle the impact of pruning from normalization and explicit plasticity injection, we compare against existing methods in the literature.

Lottery ticket baseline Frankle & Carbin (2018) argued that neural networks contain sparse sub-networks that can be trained at high levels of sparsity without gradual pruning; the authors provide an algorithm for finding these *winning tickets*. After training a network with pruning, we train a new network with the final mask *fixed* (i.e. not adjusted during training) and with the parameters initialized as in the original dense network. We found that the proposal under-performs both the pruning approach and the unpruned baseline. It is interesting to observe that both the pruning approach and the lottery ticket experiment seem to still be progressing at 40M, whereas the baseline seems to start deteriorating (Figure 13).

Dynamic sparse training baselines Evci et al. (2020) proposed RigL as a dynamic sparse training mechanism that maintains a sparse network throughout the entirety of training. In Appendix G.3 we evaluated RigL at various sparsity levels and found that, while effective, RigL is unable to

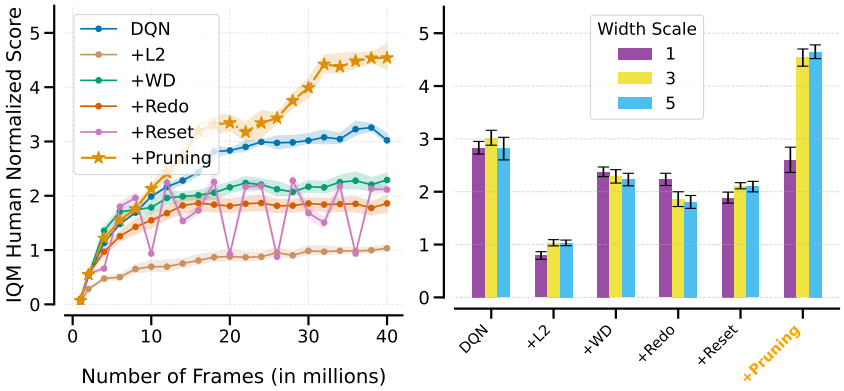

Figure 12. **Comparison against network resets (Nikishin et al., 2022), weight decay, ReDo (Sokar et al., 2023) and the normalization of (Kumar et al., 2022). Left:** Sample efficiency curves with a width factor of 3; **Right:** final performance after 40M frames with varying widths (right panel). All experiments run on DQN with the ResNet architecture and a replay ratio of 0.25.

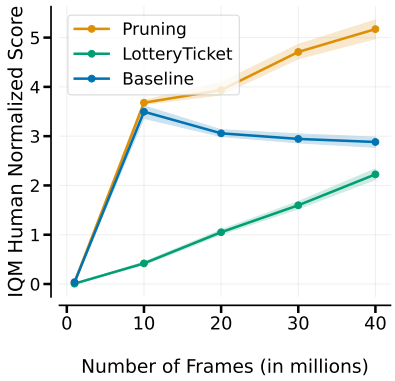

Figure 13. **Lottery ticket hypothesis experiment**. Taking the final pruned network (with a width factor of 3) and retraining with the original initialization results in worse performance.

match the performance of gradual magnitude pruning; these results are consistent with those of Graesser et al. (2022).

Normalization baselines To investigate the role normalization plays on the performance gains produced by pruning, we consider two types of ℓ_2 normalization that have proven effective in the literature. The first is weight decay (**WD**), a standard technique that adds an extra term to the loss that penalizes ℓ_2 norm of the weights, thereby discouraging network parameters from growing too large. The second is **L2**, the regularization approach proposed by Kumar et al. (2022), which is designed to enforce an ℓ_2 norm of 1 for the parameters.

Plasticity injection baselines We compare against two recent works that proposed methods for directly dealing with loss of plasticity. Nikishin et al. (2022) observed a decline in performance with an increased replay ratio, attributing it to overfitting on early samples, an effect they termed the "primacy bias". The authors suggested periodically resetting the network and demonstrated that it proved very effective at mitigating the primacy bias, and overfitting in general (this is labeled as **Reset** in our results).

Sokar et al. (2023) demonstrated that most deep RL agents suffer from the *dormant neuron phenomenon*, whereby neurons increasingly "turn off" during training of deep RL agents, thus reducing network expressivity. To mitigate this, they proposed a simple and effective method that Recycles Dormant neurons (**ReDo**) throughout training.

As Figure 12 illustrates, gradual magnitude pruning surpasses all the other regularization methods at all levels of scale, and throughout the entirety of training. Interestingly, most of the regularization methods suffer a degradation when increasing network width. This suggests that the effect of pruning cannot be solely attributed to either a form of normalization or plasticity injection. However, as we will see below, increased plasticity does seem to arise out of its use. We provide sweeps over various baseline hyperparameters in Appendices G.5 to G.8.

5.2. Impact on plasticity

Plasticity is a neural network's capacity to rapidly adjust in response to shifting data distributions (Lyle et al., 2022; 2023; Lewandowski et al., 2023); given the non-stationarity of reinforcement learning, it is crucial to maintain to ensure good performance. However, RL networks are known to lose plasticity over the course of training (Nikishin et al., 2022; Sokar et al., 2023; Lee et al., 2023).

Lyle et al. (2023) conducted an assessment of the covariance structure of gradients to examine the loss landscape of the network, and argued that improved performance, and increased plasticity, is often associated with weaker gradient correlation and reduced gradient interference. Our observations align with these findings, as illustrated in the gradient covariance heat maps in Figure 14. In dense networks, gradients exhibit a notable colinearity, whereas this colinearity is dramatically reduced in the pruned networks.

6. Discussion and Conclusion

Prior work has demonstrated that reinforcement learning agents have a tendency to under-utilize its available parameters, and that this under-utilization increases throughout training and is amplified as network sizes increase (Graesser et al., 2022; Nikishin et al., 2022; Sokar et al., 2023; Schwarzer et al., 2023). RL agents achieve strong

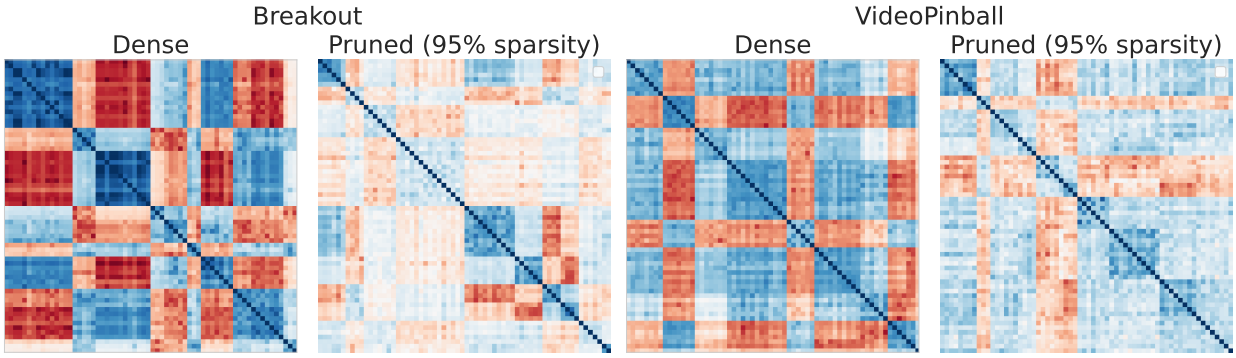

Figure 14. **Gradient covariance matrices** for Breakout (**left**) and VideoPinball (**right**) atari games. **Dark red** denotes high negative correlation, while **dark blue** indicates high positive correlation. The use of pruning induces weaker gradient correlation and less gradient interference, as evidenced by the paler hues in the heatmaps for the sparse networks.

performance in the majority of the established benchmarks with small networks (relative to those used in language models, for instance), so this evident parameter-inefficiency may be brushed off as being less critical than other, more algorithmic, considerations.

As RL continues to grow outside of academic benchmarks and into more complex tasks, it is almost surely going to necessitate larger, and more expressive, networks. In this case parameter efficiency becomes crucial to avoid the performance collapse prior works have shown, as well as for reducing computational costs (Ceron & Castro, 2021).

Our work provides convincing evidence that sparse training techniques such as gradual magnitude pruning can be effective at maximizing network utilization, especially as the initial networks are scaled (see Figures 1 and 4). The results in Figures 8, 7, and 10 all demonstrate that the sparse networks produced by pruning are better at maintaining stable performance when trained for longer. The advantages of pruning remain even when sweeping over various baseline hyper-parameters (see Appendices G.4 and G.9 to G.11). It is worth noting that the performance of the dense baselines does improve when adjusting the learning rate based on the width multiplier (Appendix G.9); however, pruning is still the most performant in these settings.

Collectively, our results demonstrate that, by meaningfully removing network parameters throughout training, we can outperform traditional dense counterparts and continue to improve performance as we grow the initial network architectures. Our results with varied agents and training regimes imply gradual magnitude pruning is a generally useful technique which can be used as a "drop-in" for maximizing agent performance.

Future work It would be natural to explore incorporating gradual magnitude pruning into recent agents that were designed for multi-task generalization (Taiga et al., 2023; Kumar et al., 2022), sample efficiency (Schwarzer et al., 2023; D'Oro et al., 2023), and generalizability (Hafner et al., 2023). Further, the observed stability of the pruned networks may have implications for methods which rely on fine-tuning or reincarnation (Agarwal et al., 2022).

Recent advances in hardware accelerators for training sparse networks may result in faster training times, and serve as an incentive for further research in methods for sparse network training. Further, the fact that a consequence of this approach is a network with fewer parameters than when initialized renders it appealing for downstream applications on edge devices.

At a minimum, we hope this work serves as an invitation to explore non-standard network architectures and topologies as an effective mechanism for maximizing the performance of reinforcement learning agents. Reinforcement learning agents typically use networks originally designed for stationary problems; therefore, other topologies might be better suited to the non-stationary nature of RL.

Acknowledgements

The authors would like to thank Laura Graesser, Utku Evci, Gopeshh Subbaraj, Evgenii Nikishin, Hugo Larochelle, Ayoub Echchahed, Zhixuan Lin and the rest of the Google DeepMind Montreal team for valuable discussions during the preparation of this work.

Laura Graesser deserves a special mention for providing us valuable feed-back on an early draft of the paper. We thank the anonymous reviewers for their valuable help in improving our manuscript. We would also like to thank the Python community (Van Rossum & Drake Jr, 1995; Oliphant, 2007) for developing tools that enabled this work, including NumPy (Harris et al., 2020), Matplotlib (Hunter, 2007), Jupyter (Kluyver et al., 2016), Pandas (McKinney, 2013) and JAX (Bradbury et al., 2018).

Impact statement

This paper presents work whose goal is to advance the field of Machine Learning, and reinforcement learning in particular. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 104–114. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/agarwal20c.html.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Reincarnating reinforcement learning: Reusing prior computation to accelerate progress. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 28955–28971. Curran Associates, Inc., 2022.

Arnob, S. Y., Ohib, R., Plis, S., and Precup, D. Single-shot pruning for offline reinforcement learning. *arXiv preprint arXiv:2112.15579*, 2021.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013. doi: 10.1613/jair.3912.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *ICML*, 2017.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588:77 – 82, 2020.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Bjorck, N., Gomes, C. P., and Weinberger, K. Q. Towards deeper deep reinforcement learning with spectral normalization. *Advances in Neural Information Processing Systems*, 34:8242–8255, 2021.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., et al. Jax: composable transformations of python+ numpy programs. 2018.

Castro, P. S., Moitra, S., Gelada, C., Kumar, S., and Bellemare, M. G. Dopamine: A research framework for deep reinforcement learning. *arXiv preprint arXiv:1812.06110*, 2018.

Ceron, J. S. O. and Castro, P. S. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *International Conference on Machine Learning*, pp. 1373–1383. PMLR, 2021.

Ceron, J. S. O., Bellemare, M. G., and Castro, P. S. Small batch deep reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=wPqEvmwFEh.

Ceron, J. S. O., Sokar, G., Willi, T., Lyle, C., Farebrother, J., Foerster, J. N., Dziugaite, G. K., Precup, D., and Castro, P. S. Mixtures of experts unlock parameter scaling for deep RL. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=X9VMhfFxwn.

Cetin, E., Ball, P. J., Roberts, S., and Celiktutan, O. Stabilizing off-policy deep reinforcement learning from pixels. In *International Conference on Machine Learning*, pp. 2784–2810. PMLR, 2022.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pp. 1096–1105. PMLR, 2018.

D'Oro, P., Schwarzer, M., Nikishin, E., Bacon, P.-L., Bellemare, M. G., and Courville, A. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OpC-9aBBVJe.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning,

I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.

Farebrother, J., Greaves, J., Agarwal, R., Lan, C. L., Goroshin, R., Castro, P. S., and Bellemare, M. G. Proto-value networks: Scaling representation learning with auxiliary tasks. In *Submitted to The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=oGDKSt9JrZi. under review.

Farebrother, J., Orbay, J., Vuong, Q., Taïga, A. A., Chebotar, Y., Xiao, T., Irpan, A., Levine, S., Castro, P. S., Faust, A., Kumar, A., and Agarwal, R. Stop regressing: Training value functions via classification for scalable deep rl. In *Forty-first International Conference on Machine Learning*. PMLR, 2024.

Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov, A., R Ruiz, F. J., Schrittwieser, J., Swirszcz, G., et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., Larochelle, H., Rowland, M., and Dabney, W. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pp. 3061–3071. PMLR, 2020.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. Noisy networks for exploration. 2018.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Graesser, L., Evci, U., Elsen, E., and Castro, P. S. The state of sparse training in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 7766–7792. PMLR, 2022.

Grooten, B., Sokar, G., Dohare, S., Mocanu, E., Taylor, M. E., Pechenizkiy, M., and Mocanu, D. C. Automatic noise filtering with dynamic sparse training in deep reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, pp. 1932–1941, Richland, SC, 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450394321.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

Hessel, M., Modayil, J., Hasselt, H. V., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. G., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018.

Hiraoka, T., Imagawa, T., Hashimoto, T., Onishi, T., and Tsuruoka, Y. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2021.

Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95, 2007.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model-based reinforcement learning for atari. *International Conference on Learning Representations*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout,

J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., and Jupyter Development Team. Jupyter Notebooks—a publishing format for reproducible computational workflows. In *IOS Press*, pp. 87–90. 2016. doi: 10.3233/978-1-61499-649-1-87.

Kostrikov, I., Yarats, D., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Kumar, A., Agarwal, R., Ghosh, D., and Levine, S. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=O9bnihsFfXU.

Kumar, A., Agarwal, R., Ma, T., Courville, A., Tucker, G., and Levine, S. Dr3: Value-based deep reinforcement learning requires explicit regularization. *arXiv preprint arXiv:2112.04716*, 2021b.

Kumar, A., Agarwal, R., Geng, X., Tucker, G., and Levine, S. Offline q-learning on diverse multi-task data both scales and generalizes. In *The Eleventh International Conference on Learning Representations*, 2022.

Lee, H., Cho, H., Kim, H., Gwak, D., Kim, J., Choo, J., Yun, S.-Y., and Yun, C. Plastic: Improving input and label plasticity for sample efficient reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Lee, J. H., Park, W., Mitchell, N. E., Pilault, J., Ceron, J. S. O., Kim, H.-B., Lee, N., Frantar, E., Long, Y., Yazdanbakhsh, A., et al. Jaxpruner: A concise library for sparsity research. In *Conference on Parsimony and Learning*, pp. 515–528. PMLR, 2024.

Lewandowski, A., Tanaka, H., Schuurmans, D., and Machado, M. C. Curvature explains loss of plasticity. *arXiv preprint arXiv:2312.00246*, 2023.

Liu, Z., Li, X., Kang, B., and Darrell, T. Regularization matters in policy optimization-an empirical study on continuous control. In *International Conference on Learning Representations*, 2020.

Livne, D. and Cohen, K. Pops: Policy pruning and shrinking for deep reinforcement learning. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):789–801, May 2020. ISSN 1941-0484. doi: 10.1109/jstsp.2020.2967566. URL http://dx.doi.org/10.1109/JSTSP.2020.2967566.

Lyle, C., Rowland, M., and Dabney, W. Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ZkC8wKoLbQ7.

Lyle, C., Zheng, Z., Nikishin, E., Pires, B. A., Pascanu, R., and Dabney, W. Understanding plasticity in neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Lyle, C., Zheng, Z., Khetarpal, K., van Hasselt, H., Pascanu, R., Martens, J., and Dabney, W. Disentangling the causes of plasticity loss in neural networks. *arXiv preprint arXiv:2402.18762*, 2024.

McKinney, W. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, 1 edition, February 2013. ISBN 9789351100065. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1449319793.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, February 2015.

Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., and Liotta, A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9 (1):2383, 2018.

Nikishin, E., Schwarzer, M., D'Oro, P., Bacon, P.-L., and Courville, A. The primacy bias in deep reinforcement learning. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16828–16847. PMLR, 17–23 Jul 2022.

Nikishin, E., Oh, J., Ostrovski, G., Lyle, C., Pascanu, R., Dabney, W., and Barreto, A. Deep reinforcement learning with plasticity injection. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=jucDLW6G9l.

Oliphant, T. E. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007. doi: 10.1109/MCSE.2007.58.

Ostrovski, G., Castro, P. S., and Dabney, W. The difficulty of passive learning in deep reinforcement learning. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=nPHA8fGicZk.

Ota, K., Jha, D. K., and Kanezaki, A. Training larger networks for deep reinforcement learning. *arXiv preprint arXiv:2102.07920*, 2021.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *CoRR*, abs/1511.05952, 2016.

Schmidt, D. and Schmied, T. Fast and data-efficient training of rainbow: an experimental study on atari. *arXiv preprint arXiv:2111.10247*, 2021.

Schmitt, S., Hudson, J. J., Zidek, A., Osindero, S., Doersch, C., Czarnecki, W. M., Leibo, J. Z., Kuttler, H., Zisserman, A., Simonyan, K., et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Schwarzer, M., Ceron, J. S. O., Courville, A., Bellemare, M. G., Agarwal, R., and Castro, P. S. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380. PMLR, 2023.

Sinha, S., Bharadhwaj, H., Srinivas, A., and Garg, A. D2rl: Deep dense architectures in reinforcement learning. *arXiv preprint arXiv:2010.09163*, 2020.

Sokar, G., Mocanu, E., Mocanu, D. C., Pechenizkiy, M., and Stone, P. Dynamic sparse training for deep reinforcement learning. *arXiv preprint arXiv:2106.04217*, 2021.

Sokar, G., Agarwal, R., Castro, P. S., and Evci, U. The dormant neuron phenomenon in deep reinforcement learning. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32145–32168. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/sokar23a.html.

Song, X., Jiang, Y., Tu, S., Du, Y., and Neyshabur, B. Observational overfitting in reinforcement learning. *arXiv preprint arXiv:1912.02975*, 2019.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, August 1988.

Taiga, A. A., Agarwal, R., Farebrother, J., Courville, A., and Bellemare, M. G. Investigating multi-task pretraining and generalization in reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Tan, Y., Hu, P., Pan, L., Huang, J., and Huang, L. RLx2: Training a sparse deep reinforcement learning model from scratch. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=DJEEqoAq7to.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirthieth AAAI Conference On Artificial Intelligence (AAAI), 2016*, 2016. cite arxiv:1509.06461Comment: AAAI 2016.

Van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Van Hasselt, H. P., Hessel, M., and Aslanides, J. When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32, 2019.

Van Rossum, G. and Drake Jr, F. L. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

Vieillard, N., Pietquin, O., and Geist, M. Munchausen reinforcement learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 4235–4246. Curran Associates, Inc., 2020.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Vischer, M., Lange, R. T., and Sprekeler, H. On lottery tickets and minimal task representations in deep reinforcement learning. In *International Conference on Learning Representations*, 2021.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pp. 1995–2003, 2016.

Yarats, D., Fergus, R., and Kostrikov, I. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

Yu, H., Edunov, S., Tian, Y., and Morcos, A. S. Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp. In *International Conference on Learning Representations*, 2019.

Zhang, C., Vinyals, O., Munos, R., and Bengio, S. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

Zhang, H., He, Z., and Li, J. Accelerating the deep reinforcement learning with neural network compression. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2019.

Zhu, M. and Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression, 2017.

A. Code availability

Our experiments were built on open source code, mostly from the Dopamine repository. The root directory for these is https://github.com/google/dopamine/tree/master/dopamine/, and we specify the subdirectories below (with clickable links):

- DQN and Rainbow agents from /jax/agents/
- Atari-100k agents from /labs/atari-100k/
- Sparsity scripts from JaxPruner /jaxpruner/baselines/dopamine/
- Resnet architecture from /labs/offline-rl/jax/networks.py **(line 108)**
- Dormant neurons metric, Reset, ReDo and Weight Decay from /labs/redo/
- SAC and PPO experiments from /google-research/rigl/rl/

For the srank metric experiments we used code from:
https://github.com/google-research/google-research/blob/master/generalization_representations_rl_aistats22/coherence/coherence_compute.py

B. Atari Game Selection

Most of our experiments were run with 15 games from the ALE suite (Bellemare et al., 2013), as suggested by Graesser et al. (2022). However, for the Atari 100k agents (subsection 4.3), we used the standard set of 26 games (Kaiser et al., 2020) to be consistent with the benchmark. We also ran some experiments with the full set of 60 games. The specific games are detailed below.

15 game subset: MsPacman, Pong, Qbert, (Assault, Asterix, BeamRider, Boxing, Breakout, CrazyClimber, DemonAttack, Enduro, FishingDerby, SpaceInvaders, Tutankham, VideoPinball. According to (Graesser et al., 2022), these games were selected to be roughly evenly distributed amongst the games ranked by DQN's human normalized score in (Mnih et al., 2015) with a lower cut off of approximately 100% of human performance.

26 game subset: Alien, Amidar, Assault, Asterix, BankHeist, BattleZone, Boxing, Breakout, ChopperCommand, CrazyClimber, DemonAttack, Freeway, Frostbite, Gopher, Hero, Jamesbond, Kangaroo, Krull, KungFuMaster, MsPacman, Pong, PrivateEye, Qbert, RoadRunner, Seaquest, UpNDown.

60 game set: The 26 games above in addition to: AirRaid, Asteroids, Atlantis, BeamRider, Berzerk, Bowling, Carnival, Centipede, DoubleDunk, ElevatorAction, Enduro, FishingDerby, Gravitar, IceHockey, JourneyEscape, MontezumaRevenge, NameThisGame, Phoenix, Pitfall, Pooyan, Riverraid, Robotank, Skiing, Solaris, SpaceInvaders, StarGunner, Tennis, TimePilot, Tutankham, Venture, VideoPinball, WizardOfWor, YarsRevenge, Zaxxon.

C. Sparsity Levels

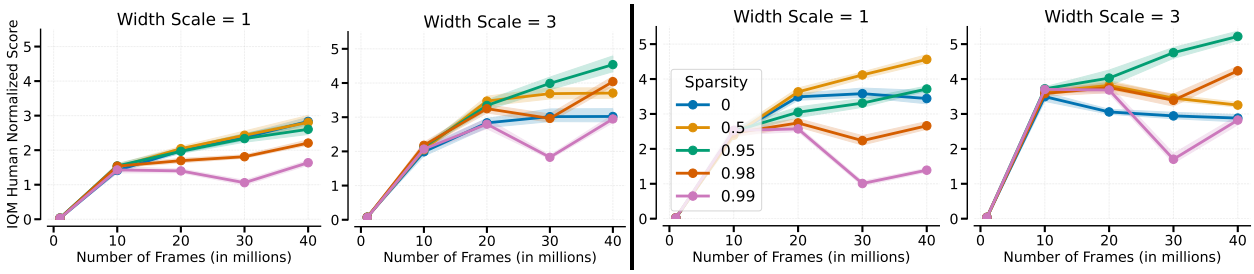

Figure 15. **Evaluating how varying the sparsity parameter affects performance** for a given architecture on DQN (Mnih et al., 2015) and Rainbow agent. We report results aggregated IQM of human-normalized scores over 15 games.

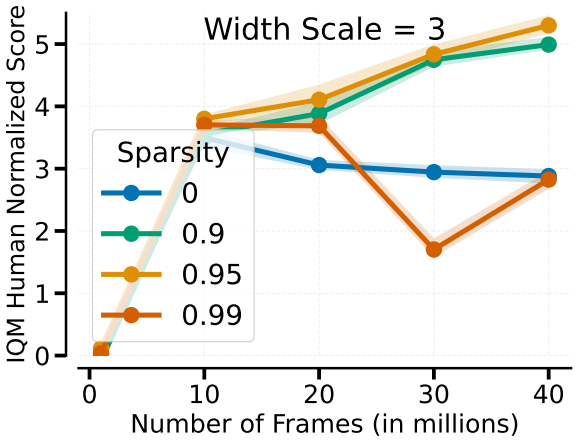

Figure 16. **Evaluating how varying the sparsity parameter affects performance** for a given architecture, resnet with a width multiplier of 3, on Rainbow agent. We report results aggregated IQM of human-normalized scores over 15 games.

D. Scaling Replay Ratios

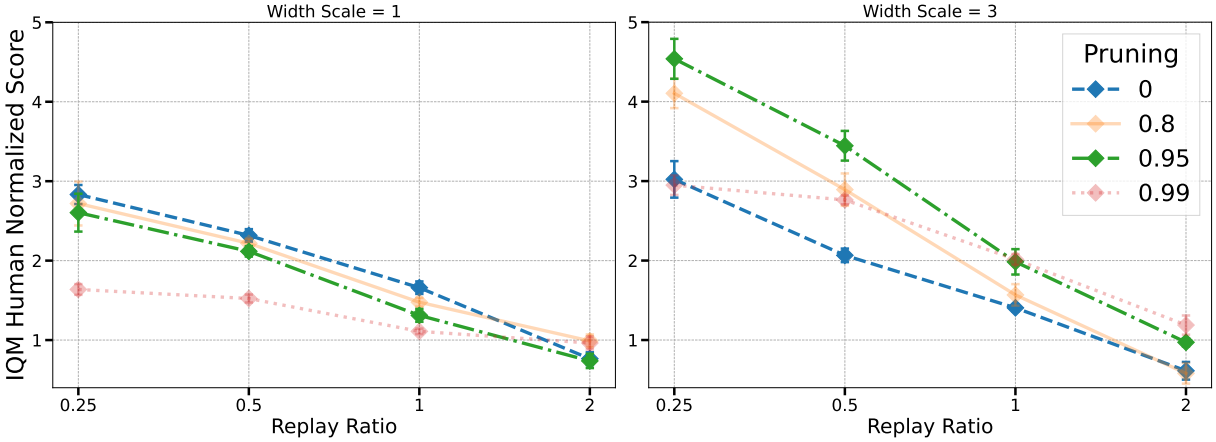

Figure 17. **Scaling replay ratio for resnet architecture** (default is 0.25), for a width factor of 1 **(left)** and a width factor of 3 **(right)** using DQN agent. We report interquantile mean performance with error bars indicating 95% confidence intervals. On the x-axis we report the replay ratio value.

E. MuJoCo environments

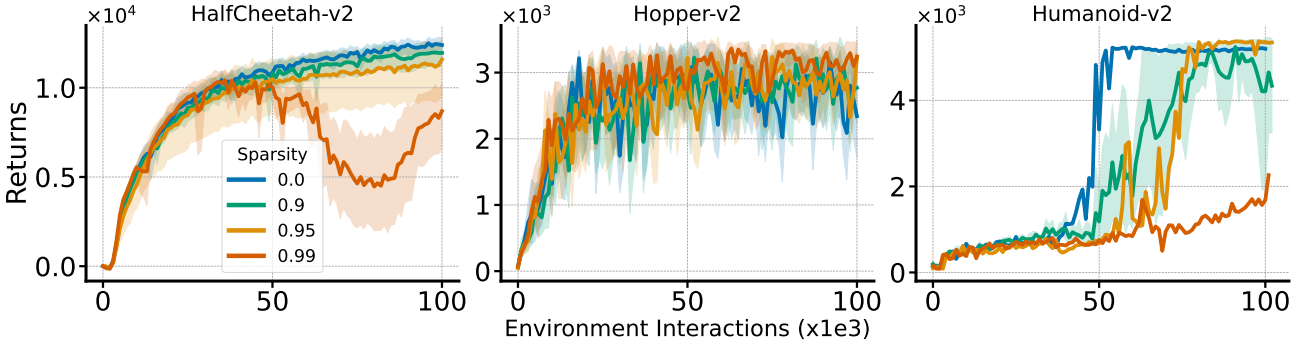

Figure 18. **Evaluating how varying the sparsity parameter affects performance of SAC** on three MuJoCo environments when increasing width x5. We report returns over 10 runs for each experiment.

F. Hyper-parameters list

Default hyper-parameter settings for DER (Van Hasselt et al., 2019) and DrQ(ϵ) (Kaiser et al., 2020; Agarwal et al., 2021). Table 1 shows the default values for each hyper-parameter across all the Atari games.

Table 1. Default hyper-parameters setting for DER and DrQ(ϵ) agents.

	Atari	
Hyper-parameter	DER	DrQ(ϵ)
Adam's(ϵ)	0.00015	0.00015
Adam's learning rate	0.0001	0.0001
Batch Size	32	32
Conv. Activation Function	ReLU	ReLU
Convolutional Width	1	1
Dense Activation Function	ReLU	ReLU
Dense Width	512	512
Normalization	None	None
Discount Factor	0.99	0.99
Exploration ϵ	0.01	0.01
Exploration ϵ decay	2000	5000
Minimum Replay History	1600	1600
Number of Atoms	51	0
Number of Convolutional Layers	3	3
Number of Dense Layers	2	2
Replay Capacity	1000000	1000000
Reward Clipping	True	True
Update Horizon	10	10
Update Period	1	1
Weight Decay	0	0
Sticky Actions	False	False

Default hyper-parameter settings for DQN (Mnih et al., 2015), Rainbow (Hessel et al., 2018), IQN (Dabney et al., 2018), Munchausen-IQN (Vieillard et al., 2020). Table 2 shows the default values for each hyper-parameter across all the Atari games.

Table 2. Default hyper-parameters setting for DQN, Rainbow, IQN, Munchausen-IQN agents.

	Atari			
Hyper-parameter	DQN	Rainbow	IQN	M-IQN
Adam's (ϵ)	1.5e-4	1.5e-4	3.125e-4	3.125e-4
Adam's learning rate	6.25e-5	6.25e-5	5e-5	5e-5
Batch Size	32	32	32	32
Conv. Activation Function	ReLU	ReLU	ReLU	ReLU
Convolutional Width	1	1	1	1
Dense Activation Function	ReLU	ReLU	ReLU	ReLU
Dense Width	512	512	512	512
Normalization	None	None	None	None
Discount Factor	0.99	0.99	0.99	0.99
Exploration ϵ	0.01	0.01	0.01	0.01
Exploration ϵ decay	250000	250000	250000	250000
Minimum Replay History	20000	20000	20000	20000
Number of Atoms	0	51	-	-
Kappa	-	-	1.0	1.0
Num tau samples	-	-	64	64
Num tau prime samples	-	-	64	64
Num quantile samples	-	-	32	32
Number of Convolutional Layers	3	3	3	3
Number of Dense Layers	2	2	2	2
Replay Capacity	1000000	1000000	1000000	1000000
Reward Clipping	True	True	True	True
Update Horizon	1	3	3	3
Update Period	4	4	4	4
Weight Decay	0	0	0	0
Sticky Actions	True	True	True	True
Tau	-	-	0	0.03

Default hyper-parameter settings for CQL (Kumar et al., 2020) and CQL+C51 (Kumar et al., 2022) offline agents. Table 3 shows the default values for each hyper-parameter across all the Atari games.

Table 3. Default hyper-parameters setting for CQL and CQL+C51 agents.

	Atari	
Hyper-parameter	CQL	CQL+C51
Adam's(ϵ)	0.0003125	0.00015
Batch Size	32	32
Conv. Activation Function	ReLU	ReLU
Convolutional Width	1	1
Dense Activation Function	ReLU	ReLU
Normalization	None	None
Dense Width	512	512
Discount Factor	0.99	0.99
Learning Rate	0.00005	0.0000625
Number of Atoms	0	51
Number of Convolutional Layers	3	3
Number of Dense Layers	2	2
Fixed Replay Capacity	2,500,000 steps	2,500,000 steps
Reward Clipping	True	True
Update Horizon	1	3
Update Period	1	1
Weight Decay	0	0
Replay Scheme	Uniform	Uniform
Dueling	False	True
Double DQN	False	True
CQL coef	0.1	0.1

Default hyper-parameter settings for CNN architecture (Mnih et al., 2015) and Impala-based ResNet (Espeholt et al., 2018) Table 4 shows the default values for each hyper-parameter across all the Atari games.

Table 4. Default hyper-parameters for neural networks.

	Atari	
Hyper-parameter	CNN architecture (Mnih et al., 2015)	Impala-based ResNet (Espeholt et al., 2018)
Observation down-sampling	(84, 84)	(84, 84)
Frames stacked	4	4
Q-network (channels)	32, 64, 64	32, 64, 64
Q-network (filter size)	8 x 8, 4 x 4, 3 x 3	8 x 8, 4 x 4, 3 x 3
Q-network (stride)	4, 2, 1	4, 2, 1
Num blocks	-	2
Use max pooling	False	True
Skip connections	False	True
Hardware	Tesla P100 GPU	Tesla P100 GPU

G. Additional experiments

Unless otherwise specified, in all experiments below we report the interquantile mean after 40 million environment steps, aggregated over 15 games with 5 seeds each; error bars indicate 95% stratified bootstrap confidence intervals (Agarwal et al., 2021).

G.1. Experiments with PPO on MuJoCo

We used the PPO implementation from (Graesser et al., 2022) and ran some initial experiments with MuJoCo, increasing the width by 5x. As with our SAC experiments, we see no real change in performance, with perhaps some mild gains in Humanoid-v2. One reason why we may not see performance improvements in neither SAC nor PPO is that in ALE experiments, all agents use Convolutional layers, whereas for the MuJoCo experiments (where we ran SAC and PPO) the networks only use dense layers.

Nonetheless, it is worth noting that Graesser et al. (2022) saw degradation with pruning at width=1 (Figure 16 in their paper), with almost a total collapse at 99% sparsity. In contrast, our results with 5x width shows strong performance even at 99% sparsity.

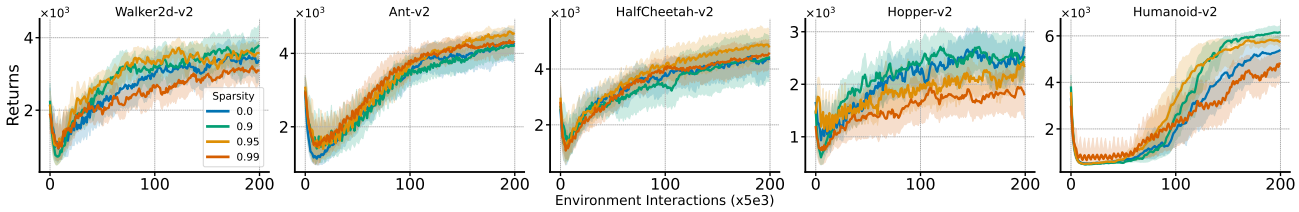

Figure 19. **Proximal Policy Optimization (PPO)** (Schulman et al., 2017) on MuJoCo environments when increasing width x5. We report returns over 10 runs for each experiment.

G.2. Experiments with IQN and M-IQN

While Rainbow is still a competitive agent in the ALE and both DQN and Rainbow are still regularly used as baselines in recent works, exploring newer agents is a reasonable request. To address this, we ran experiments with Implicit Quantile Networks (IQN) (Dabney et al., 2018) and Munchausen-IQN (Vieillard et al., 2020) with widths of 1 and 3; consistent with our submission's findings, we observe significant gains when using pruning.

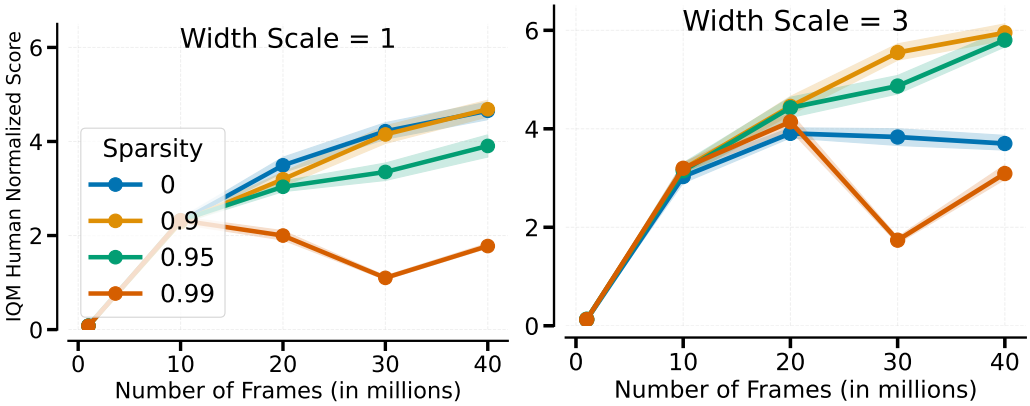

Figure 20. **IQN (*Implicit Quantile Networks*)** (Dabney et al., 2018) with ResNet architecture (with a width factor of 1 and 3).

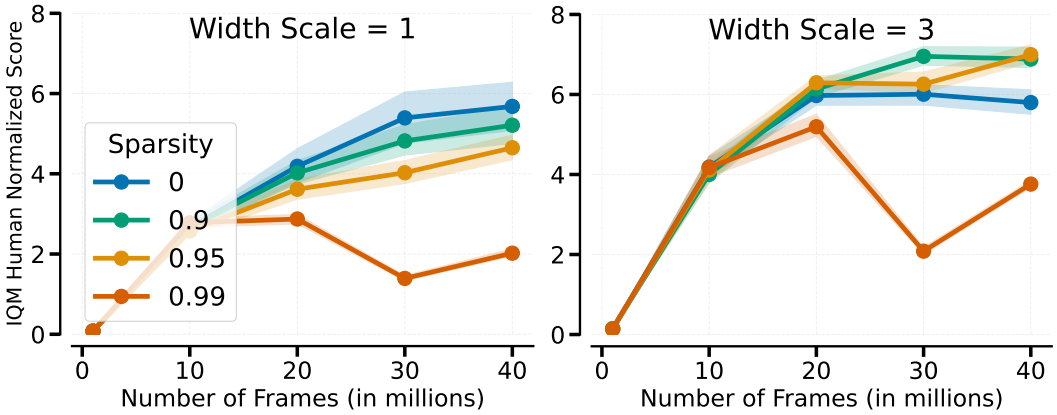

Figure 21. **M-IQN (*Munchausen-Implicit Quantile Networks*)** (Vieillard et al., 2020) with ResNet architecture (with a width factor of 1 and 3).

G.3. Comparison with RigL

We have run a comparison with RigL (Evci et al., 2020). While RigL can be somewhat effective, it is unable to match the performance of pruning.

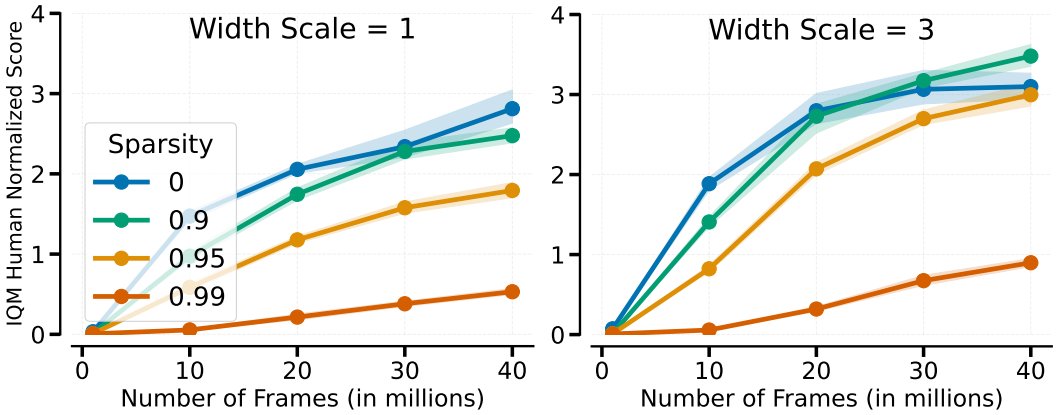

Figure 22. **The Rigged Lottery (RigL)** (Evci et al., 2020) for DQN with ResNet architecture (with a width factor of 1 and 3).

G.4. Varying Adam's ϵ

The default value for Adam's ϵ is $1.5e-5$; we ran experiments by dividing/multiplying this value by 3 ($5e-5$ and $4.5e-4$, respectively). In all these cases, pruning maintains a significant advantage over the dense baseline.

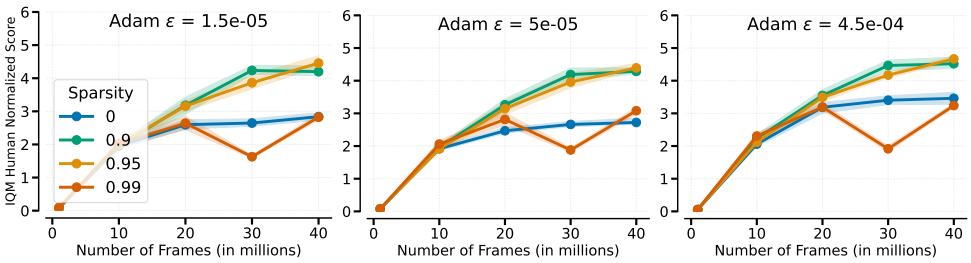

Figure 23. **Adam's epsilon (ϵ)** (Kingma & Ba, 2014) for DQN with ResNet architecture and a width multiplier of 3.

G.5. Sweeping over ReDo threshold τ

This parameter (introduced in Definition 3.1 of (Sokar et al., 2023)) defines the threshold for determining neuron dormancy. Sokar et al. (2023) suggested using 0.1 with the CNN network. Since we are using the Impala network architecture, we tested three additional values: (0, 0.025, 0.3). We found that 0.1, as used in Figure 12 of our submission, yields the best performance.

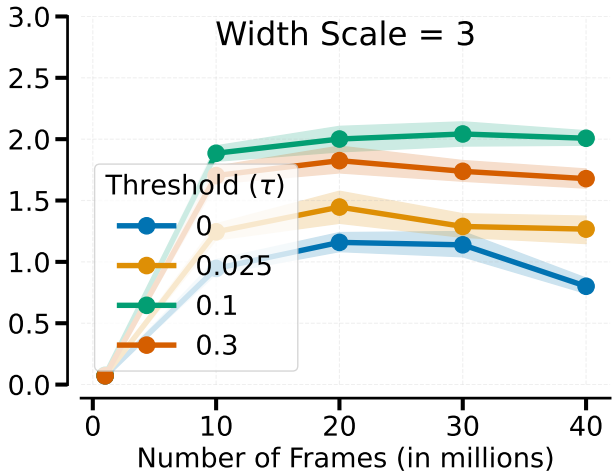

Figure 24. **Varying ReDo's τ threshold** (Sokar et al., 2023) for DQN with ResNet architecture and a width multiplier of 3.

G.6. Frequency of network resets

We varied the frequency of network resets (Nikishin et al., 2022) to evaluate whether this could help mitigate the performance loss when increasing the network width. While more infrequent resets (every 250000 steps compared to the default value of 100000) improves performance slightly, it still drastically under-performs with respect to the baseline and the pruning approach.

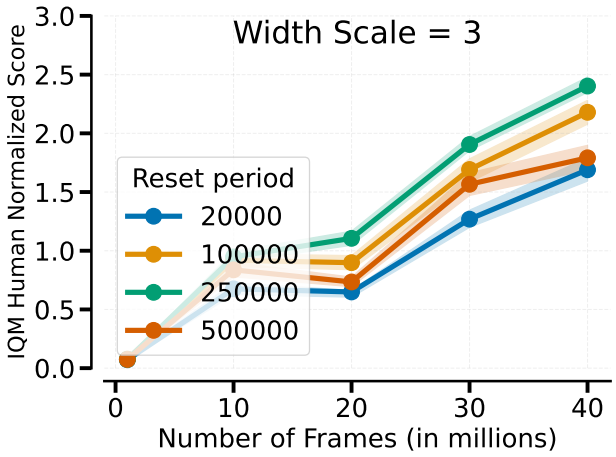

Figure 25. **Varying the reset period** (Nikishin et al., 2022) for DQN with ResNet architecture and a width multiplier of 3.

G.7. Layer to reset

In our paper we followed the approach of Nikishin et al. (2022) and Sokar et al. (2023) of resetting only the last layer. We explored resetting different layers, but found it resulted in no significant performance difference.

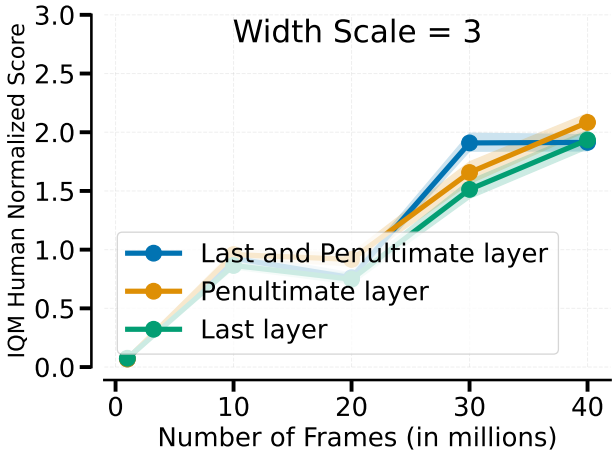

Figure 26. **Resetting different layers** (Nikishin et al., 2022), DQN with ResNet architecture and a width multiplier of 3.

G.8. Varying weight decay

We ran a sweep over the following values $10e^{-6}$, $10e^{-5}$, $10e^{-4}$, $10e^{-3}$, and $10e^{-2}$, using the Impala architecture with a width factor of 3. The best performance is obtained with $10e^{-5}$, which is the value suggested by Sokar et al. (2023), and the value used in Figure 12 of our submission.

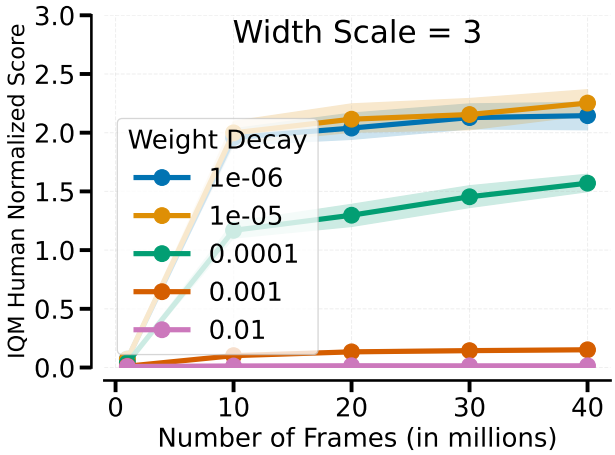

Figure 27. **Weight Decay (WD)** for DQN with ResNet architecture with a width factor of 3.

G.9. Varying learning rates

The default learning for DQN is $6.25e-5$. As suggested by the reviewer, we have run experiments with a learning rate divided by the width scale factor (so $2.08e-5$ for a width factor of 3, and $1.25e-5$ for a width factor of 5). These learning rates do improve the performance of the baseline, but it is still surpassed by pruning. These results are consistent with the thesis of the paper: *pruning can serve as a drop-in mechanism for increasing agent performance.*

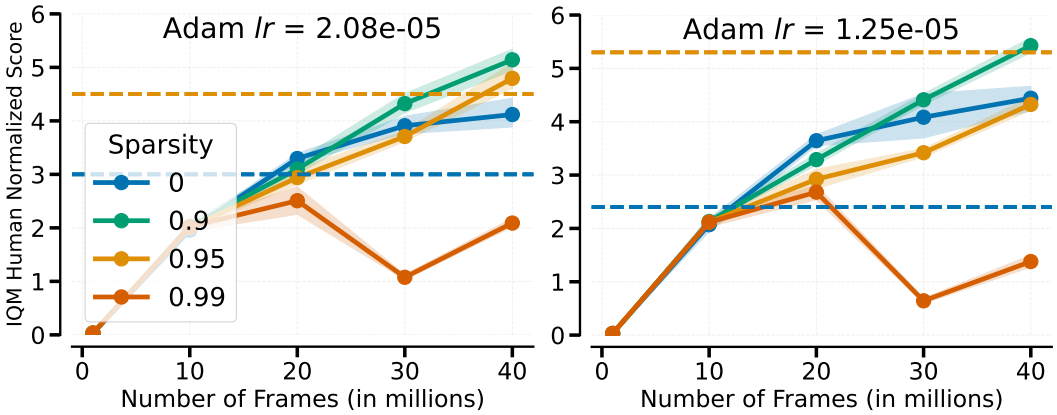

Figure 28. **Learning rate evaluation** for DQN agent with ResNet architecture (with a width factor of 3 **left** and 5 **right**). These learning rate values correspond to dividing the default learning rate by the factor we used to amplify the size of the neural network. The dashed lines indicate the final performance for dense (blue) and 0.95% sparse (orange) nets when using the default learning rate (*lr:* $6.25e-5$).

G.10. Varying batch size

The default batch size is 32, and ran experiments with batch sizes of 16 and 64. In all cases, pruning maintains its strong advantage.

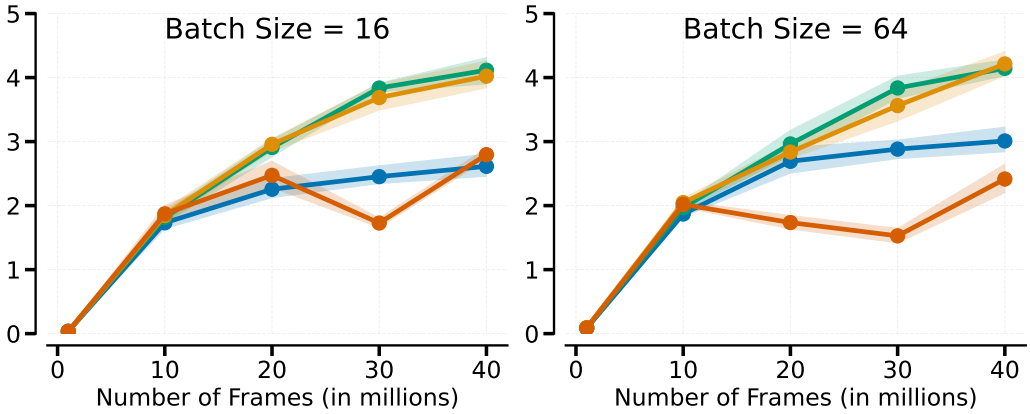

Figure 29. **Batch size** (Ceron et al., 2023) for DQN with ResNet architecture and a width multiplier of 3.

G.11. Varying update horizon

We explored using an update horizon of 3 for DQN (the default is 1) and found that pruning still maintains its advantage.

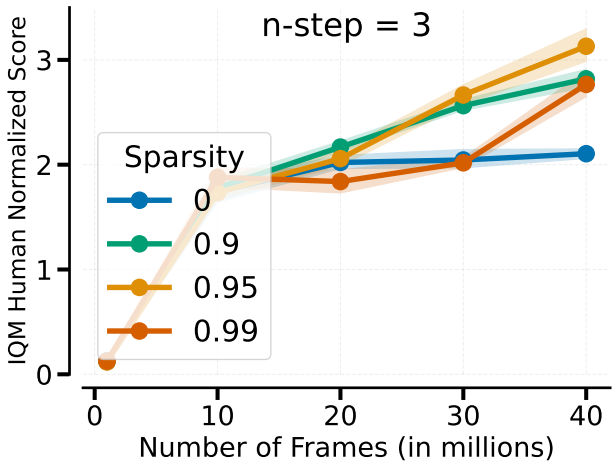

Figure 30. **Multi-step return** (Sutton, 1988) for DQN with ResNet architecture and a width multiplier of 3.